Finally, the reconstructed sample residual values X'' from Equation 8-59 are added to the prediction values P_{ij} from motion compensated prediction or spatial prediction and clipped to the range of 0 to 255 to form the final decoded sample result prior to application of the deblocking filter as specified in subclause 8.~~7~~6.3.

12.4.4 Modifications for the deblocking filter

If ABT is used, the boundary strength shall be Bs = 0 for all 4x4 luma block edges inside an ABT block. The index into the threshold table (Table 8-3) is increased by I_{QP}. For ABT blocks, I_{QP} depends on the sizes of the neighbouring blocks as specified in Table 12-4. I_{QP} = 0 for non-ABT blocks.

Table 12-4 – I_{QP} values

I_{QP}		Block q			
		4x4	4x8	8x4	8x8
Block p	4x4	0	1	1	2
	4x8	1	2	2	3
	8x4	1	2	2	3
	8x8	2	3	3	3

The index used to access the α-table, as well as the C0-table that is used in the default filter mode, is computed as:

$$\text{Index}_A = \text{Clip3}(0, 51, QP_{av} + I_{QP} + \text{Filter\_Offset\_A})$$

The index used to access the β-table is computed as:

$$\text{Index}_B = \text{Clip3}(0, 51, QP_{av} + I_{QP} + \text{Filter\_Offset\_B}),$$

with QP_{av}, Filter_Offset_A, and Filter_Offset_B as specified in subclause 8.7.2. The values for the thresholds (α and β) are specified in Table 8-3.

12.5 ABT entropy coding

12.5.1 ABT variable length coding

12.5.1.1 Mapped Exp-Golomb entropy coding

The ABT intra macroblock types in Intra slices and the intra_block_typeABT syntax elements in Inter slices are to Exp-Golomb codeword numbers as specified in Table 12-5.

Table 12-5 – Assignment of Exp-Golomb codeword numbers for ABT syntax elements

code_number	mb_type	intra_block_typeABT
0	ABTIntra_4x4	4x4
1	ABTIntra_4x8	4x8
2	ABTIntra_8x4	8x4
3	ABTIntra_8x8	8x8

12.5.1.2 VLC entropy coding of ABT coefficients

12.5.1.2.1 Decoding ~~num_coeffABT~~num_coeff_abt

For ABT Intra blocks, ~~num_coeffABT~~num_coeff_abt is specified. The structure of the codewords for num_coeff and escape_run, specified in subclause 12.4.2.4, is indicated in Table 12-6. The info bits x_i, i=0 to n can take values 0 or 1.

Table 12-6 – Code structure for ABT ~~num_coeffABT~~num_coeff_abt and escape_run

codeword	length L
1 x1 x0	3
0 1 x2 x1 x0	5
0 0 1 x3 x2 x1 x0	7
0 0 0 1 x4 x3 x2 x1 x0	9
0 0 0 0 1 x5 x4 x3 x2 x1 x0	11

The value of num_coeff is specified as the code number of the decoded codeword. The code number is specified as

$$\text{code\_number} = 2^{(L+1)/2} - 4 + \text{INFO} \qquad (12\text{-}13)$$

For a codeword with info bits xi, i=0 to n, INFO is specified as

$$\text{INFO} = \sum_{i=0}^{n} \text{x}i \cdot 2^{i}\,. \qquad (12\text{-}14)$$

12.5.1.2.2 2D (level,run) symbols

The code structure used for decoding (level,run) symbols depends on block type. The structure of the codes is specified in Table 12-7. For all block types, the codewords with code numbers 0 to 59 are used, with code number 59 being the escape symbol. INFO is specified in Equation 12-14

Table 12-7 – Code structure for ABT (level, run) symbols

Block type	Code structure	length L	code_number
Intra 8x8, 8x4, 4x8, 4x4 Inter 8x8	1 x1 x0	3	$2^{(L+2)/2} - 4 + \text{INFO}$
	0 1 x2 x1 x0	5	
	0 0 1 x3 x2 x1 x0	7	
	0 0 0 x4 x3 x2 x1 x0	8	
Inter 8x4, 4x8	1 x0	2	$2^{(L+1)/2} - 2 + \text{INFO}$
	0 1 x1 x0	4	
	0 0 1 x2 x1 x0	6	
	0 0 0 1 x3 x2 x1 x0	8	
	0 0 0 0 x4 x3 x2 x1 x0	9	
Inter 4x4	1	1	0
	0 1 x0	3	$2^{L/2} - 1 + \text{INFO}$
	0 0 1 x1 x0	5	
	0 0 0 1 x2 x1 x0	7	
	0 0 0 0 1 x3 x2 x1 x0	9	
	0 0 0 0 0 x4 x3 x2 x1 x0	10	$2^{(L+1)/2} - 1 + \text{INFO}$

12.5.1.2.3 Assignment of level and run to code numbers

For positive level, the assignment of code numbers to run and level is specified in Table 12-9. Run and negative levels are assigned as follows

$$\text{code\_number(-abs(level), run) = code\_number(abs(level), run) + 1.} \qquad (12\text{-}15)$$

12.5.1.2.4 escape_level and escape_run

The code structure for escape_level is specified in Table 12-8. The code number of a decoded codeword is

$$\text{code\_number} = 2^{(L+2)/2} - 8 + \text{INFO}, \qquad (12\text{-}16)$$

with INFO as specified in Equation 12-14. The assignment of code numbers to escape_level is specified as follows:

Formatted: Equation

```
if( ( code_number % 2) > 0 )
  escape_level = -(code_number/2)
else
  escape_level = code_number/2                                   (12-17)
```

The code structure for escape_run is specified in Table 12-6. The value of escape_run is specified as the code number of the decoded codeword. The code number for escape_run decoding is specified in ~~Table~~ Equation 12-13.

Table 12-8 – Code structure for escape_level

Codeword	length L
1 x2 x1 x0	4
0 1 x3 x2 x1 x0	6
0 0 1 x4 x3 x2 x1 x0	8
0 0 0 1 x5 x4 x3 x2 x1 x0	10
0 0 0 0 1 x6 x5 x4 x3 x2 x1 x0	12
0 0 0 0 0 1 x7 x6 x5 x4 x3 x2 x1 x0	14
0 0 0 0 0 0 1 x8 x7 x6 x5 x4 x3 x2 x1 x0	16
0 0 0 0 0 0 0 1 x9 x8 x7 x6 x5 x4 x3 x2 x1 x0	18

Table 12-9 – Assignment of Inter and Intra level and run to code numbers.

run	Inter								Intra																				
	EOB	level>0							level>0 (QP < 26)							level>0 (26 <= QP < 34)							level>0 (QP >= 34)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
0	0	1	5	13	21	31	39	47	0	2	6	8	12	18	20	0	2	8	12	18	22	30	0	4	12	20	32	42	56
1	-	3	15	33	51	-	-	-	4	14	24	30	38	44	49	4	16	26	36	48	56	-	2	16	30	48	-	-	-
2	-	7	25	53	-	-	-	-	10	28	40	50	-	-	-	6	24	42	-	-	-	-	6	24	46	-	-	-	-
3	-	9	35	-	-	-	-	-	16	34	54	-	-	-	-	10	34	-	-	-	-	-	8	34	-	-	-	-	-
4	-	11	45	-	-	-	-	-	22	42	-	-	-	-	-	16	44	-	-	-	-	-	10	40	-	-	-	-	-
5	-	17	55	-	-	-	-	-	26	56	-	-	-	-	-	20	5	-	-	-	-	-	14	50	-	-	-	-	-
6	-	19	-	-	-	-	-	-	32	-	-	-	-	-	-	28	-	-	-	-	-	-	18	-	-	-	-	-	-
7	-	23	-	-	-	-	-	-	36	-	-	-	-	-	-	32	-	-	-	-	-	-	22	-	-	-	-	-	-
8	-	27	-	-	-	-	-	-	46	-	-	-	-	-	-	38	-	-	-	-	-	-	26	-	-	-	-	-	-

9	-	29	-	-	-	-	-	-	52	-	-	-	-	-	-	40	-	-	-	-	-	-	28	-	-	-	-	-	-
10	-	37	-	-	-	-	-	-	-	-	-	-	-	-	-	46	-	-	-	-	-	-	36	-	-	-	-	-	-
11	-	41	-	-	-	-	-	-	-	-	-	-	-	-	-	52	-	-	-	-	-	-	38	-	-	-	-	-	-
12	-	43	-	-	-	-	-	-	-	-	-	-	-	-	-	54	-	-	-	-	-	-	44	-	-	-	-	-	-
13	-	49	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	52	-	-	-	-	-	-
14	-	57	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	54	-	-	-	-	-	-

12.5.2 ABT CABAC

12.5.2.1 Fixed-length (FL) binarization for mb_type

Table 12-10 shows the binarization scheme used for decoding of macroblock types in I-slices. In case of mb_type = 6 for P slices or mb_type = 23 for B slices, intra_block_typeABT has to be decoded as additional information related to the chosen intra mode for these macroblock types. This shall be done in the same way as specified for mb_type in I slices.

Table 12-10 – Binarization for macroblock type

Slice type	Code number for mb_type	Binarization							
I slice	0 (ABTIntra_4x4)	1	0						
	1 (ABTIntra_4x8)	0	0						
	2 (ABTIntra_8x4)	0	1						
	3 (ABTIntra_8x8)	1	1						

12.5.2.2 Context definition and assignment

Table 12-11 provides the context identifier associated to the syntax element macroblock type. A detailed description of the corresponding context variables is given in the subsequent subclauses. For the syntax elements related to decoding of transform coefficients, each of the context identifiers utilizes a separate set of ranges depending on whether the additional context categories 5 – 7 given in Table 12-14 are used, which is only the case if MbABTFlag=1.

Table 12-11 – Macroblock type and associated context identifier

Syntax element	Context identifier	Type of Binarization	max_idx_ctx_id	Range of context label
Macroblock type	ctx_mb_type_I_ABT	Table 10-16	2	0 – 4
Transform coefficients	ctx_cbp4	-/-	-/-	75 – 94, 267 – 274
	ctx_sig	-/-	-/-	95 – 155, 275 – 319
	ctx_last	-/-	-/-	156 – 216, 320 – 346
	ctx_abs_level	UEG0, UCoff=14	2	217 – 266, 347 – 376

12.5.2.2.1 Assignment of context labels

Tables 12-12 and 12-13 contain context identifiers along with their corresponding range of context labels. The association of context labels (modulo some offset) and bin numbers shows which context variable uses a fixed model and which one implies a choice of different models. The latter are characterized by those entries where a set of different context labels are given for a specific bin_no (Table 12-12) or block type dependent context_category (Table 12-13); these are the context variables, which need to be specified further in the following subclauses 12.5.2.2.2 and 12.5.2.2.3.

Table 12-12 – Context identifiers and associated context labels

Context identifier	Range of context label	Offset for context label	max_idx_ctx_id	bin_no				
				1	2	3	4	≥ 5
ctx_mb_type_I_ABT	0 – 4	0	2	0,1,2	3,4	-/-	-/-	-/-
ctx_abs_level (context_category 5 – 7)	347 – 376	347+ 10*(context_category–5)	2	0 to 4	5 to 9	5 to 9	5 to 9	5 to 9

Table 12-13 – Context identifiers and associated context labels (continued)

Context identifier	Offset (range) of context label for context_category 5 – 7	context_category of block_type							
		0	1	2	3	4	5	6	7
ctx_cbp4	267 (267 – 274)	0 – 3	4 – 7	8 – 11	12 – 15	16 – 19	-/-	0 – 3	4 – 7
ctx_sig	275 (275 – 319)	0 – 14	15 – 28	29 – 43	44 –46	47 – 60	0 – 14	15 –29	30 – 44
ctx_last	320 (320 – 346)	0 – 14	15 – 28	29 – 43	44 –46	47 – 60	0 – 9	10 –19	20 – 29

12.5.2.2.2 Context definitions using preceding bin values

For the context identifier ctx_mb_type_I_ABT, the choice of the model for the 2nd bin depends on the value of the first bin as specified in Equation 12-18:

$$\text{ctx\_mb\_type\_I\_ABT}[2] = (b1 == 0)\ ?\ 3 : 4 \qquad (12\text{-}18)$$

Table 12-14 – Additional context categories for the different block types

block_type	Maximum number of coefficients	context_category
Luma block for INTRA 8x8 mode	64	5:Luma-8x8
Luma block for INTER 8x8 mode	64	
Luma block for INTRA 8x4 mode	32	6:Luma-8x4
Luma block for INTER 8x4 mode	32	
Luma block for INTRA 4x8 mode	32	7:Luma-4x8
Luma block for INTRA 4x8 mode	32	

12.5.2.2.3 Additional context definitions for information related to transform coefficients

As specified in subclause 9.2.2.4, three different additional context identifiers are used for conditioning of information related to transform coefficients. All these three types depend on context categories of different block types denoted by the variable *context_category*. The definition of these context categories is given in Tables 9-24 and 12-14. Note that the context categories 5 – 7 are only used in the case when MbABTFlag = 1. The context identifiers ctx_sig and ctx_last are related to the binary valued information of SIG and LAST; the definition of the related context variables includes an additional dependency on the scanning position *scanning_pos* within the regarded block:

$$ctx\_sig[scanning\_pos] = Map\_sig(scanning\_pos), \qquad (12\text{-}19)$$

$$ctx\_last[scanning\_pos] = Map\_last(scanning\_pos). \qquad (12\text{-}20)$$

The definition of Map_sig and Map_last in Equations 12-19 and 12-20 depends on the block type. For context_category 0 – 4 the corresponding maps are given by

$$Map\_sig(scanning\_pos) = Map\_list(scanning\_pos) = scanning\_pos, \quad \text{if } context\_category = 0,\ldots,4, \qquad (12\text{-}21)$$

where scanning_pos denotes the position related to the zig-zag scan. For context categories 5 – 7, which are only in use if MbABTFlag = 1, two cases are distinguished. In frame coding mode, where the zig-zag scan is used, Map_sig and Map_last are given by the definition in Table 12-15; for field coding mode, Map_sig and Map_last related to the alternative scans are given in Tabel 12-16. In each case, the offset for the context category given in Table 12-17 has to be added for calculating the context label of each scanning position.

For abs_level_m1, the decoding process is specified in subclause 9.2.2.4.

Table 12-15 – *Map_sig* and *Map_last* for zig-zag scanning order used for the additional ABT block sizes 8x8, 8x4 and 4x8

Scanning position	8x8		Scanning position	8x8		Scanning position	8x4 and 4x8	
	Map_sig	Map_last		Map_sig	Map_last		Map_sig	Map_last
0	0	0	32	7	3	0	0	0
1	1	1	33	6	3	1	1	1
2	2	1	34	11	3	2	2	1
3	3	1	35	12	3	3	3	1
4	4	1	36	13	3	4	4	1
5	5	1	37	11	3	5	5	1
6	5	1	38	6	3	6	7	1
7	4	1	39	7	3	7	8	1
8	4	1	40	8	4	8	9	2
9	3	1	41	9	4	9	10	2
10	3	1	42	14	4	10	11	2
11	4	1	43	10	4	11	9	2
12	4	1	44	9	4	12	8	2
13	4	1	45	8	4	13	6	2
14	5	1	46	6	4	14	7	2
15	5	1	47	11	4	15	8	2
16	4	2	48	12	5	16	9	3
17	4	2	49	13	5	17	10	3
18	4	2	50	11	5	18	11	3
19	4	2	51	6	5	19	9	3
20	3	2	52	9	6	20	8	4
21	3	2	53	14	6	21	6	4
22	6	2	54	10	6	22	12	4
23	7	2	55	9	6	23	8	4

24	7	2	56	11	7	24	9	5
25	7	2	57	12	7	25	10	5
26	8	2	58	13	7	26	11	6
27	9	2	59	11	7	27	9	6
28	10	2	60	14	8	28	13	7
29	9	2	61	10	8	29	13	7
30	8	2	62	12	8	30	14	8
31	7	2	63			31		

Table 12-16 – *Map_sig* and *Map_last* for field-based scanning order used for the additional ABT block sizes 8x8, 8x4 and 4x8

Scanning position	8x8		Scanning position	8x8		Scanning position	8x4 and 4x8	
	Map_sig	Map_last		Map_sig	Map_last		Map_sig	Map_last
0	0	0	32	9	3	0	0	0
1	1	1	33	9	3	1	1	1
2	1	1	34	10	3	2	2	1
3	2	1	35	10	3	3	3	1
4	2	1	36	8	3	4	4	1
5	3	1	37	11	3	5	5	1
6	3	1	38	12	3	6	6	1
7	4	1	39	11	3	7	3	1
8	5	1	40	9	4	8	4	2
9	6	1	41	9	4	9	5	2
10	7	1	42	10	4	10	6	2
11	7	1	43	10	4	11	3	2
12	7	1	44	8	4	12	4	2
13	8	1	45	13	4	13	7	2
14	4	1	46	13	4	14	6	2
15	5	1	47	9	4	15	8	2
16	6	2	48	9	5	16	9	3
17	9	2	49	10	5	17	7	3
18	10	2	50	10	5	18	6	3
19	10	2	51	8	5	19	8	3
20	8	2	52	13	6	20	9	4
21	11	2	53	13	6	21	10	4
22	12	2	54	9	6	22	11	4
23	11	2	55	9	6	23	12	4

24	9	2	56	10	7	24	12	5
25	9	2	57	10	7	25	10	5
26	10	2	58	14	7	26	11	6
27	10	2	59	14	7	27	13	6
28	8	2	60	14	8	28	13	7
29	11	2	61	14	8	29	14	7
30	12	2	62	14	8	30	14	8
31	11	2	63			31		

12.5.2.3 Initialisation of context models

The initialization procedure for the context models is specified in subclause 9.2.3. In this subclause, the initialization parameters for the additional context models in subclause 12.4.2 are specified.

Table 12-17 – Initialisation parameters for context identifier *ctx_mb_type_I_ABT*

Context label	ctx_mb_type_I_ABT	
	m	n
0	-8	53
1	2	50
2	17	20
3	2	50
4	2	50

Table 12-18 – Initialisation parameters for context identifiers *ctx_cbp4, ctx_sig, ctx_last, ctx_abs_level* for context category 5 – 7

Context label	Context category 5				Context label	Context category 6				Context label	Context category 7			
	I-slices		P,B-slices			I-slices		P,B-slices			I-slices		P,B-slices	
	m	n	m	n		m	n	m	n		m	n	m	n
ctx_cbp4														
					267	-4	63	-2	61	271	-1	63	-3	63
					268	-1	70	-7	75	272	-7	74	-15	75
					269	-7	68	-12	70	273	-1	70	-13	80
					270	-5	76	-20	86	274	-5	76	-21	88
ctx_sig														
275	-1	59	-4	44	290	-8	69	-3	48	305	-8	68	-3	49
276	-7	55	-3	34	291	-12	67	-5	47	306	-11	68	-6	47
277	-6	58	-2	35	292	-11	68	-3	47	307	-11	64	-3	48
278	-7	53	-4	33	293	-12	63	-7	47	308	-11	56	-7	46
279	-9	52	-5	31	294	-12	66	-7	48	309	-13	63	-6	47
280	-5	48	-2	31	295	-12	66	-3	46	310	-11	67	-3	45

281	-14	59	-7	36	296	-7	60	-7	48	311	-8	63	-8	49
282	-8	53	-1	31	297	-12	63	-4	45	312	-12	65	-6	46
283	-10	54	-4	33	298	-11	64	-2	45	313	-11	63	-3	46
284	-5	47	4	29	299	-14	64	-5	46	314	-13	60	-2	45
285	-4	43	0	29	300	-12	57	-8	43	315	-11	53	-7	45
286	-13	56	2	31	301	-16	57	0	35	316	-13	52	-2	37
287	-9	49	0	31	302	-9	53	5	40	317	-8	50	0	38
288	-8	47	5	23	303	7	54	-3	57	318	8	52	-3	54
289	3	44	15	28	304	2	67	5	63	319	1	67	7	64
ctx_last														
320	12	29	17	27	329	9	25	24	10	338	8	25	25	9
321	5	29	23	17	330	7	25	25	7	339	8	24	25	7
322	9	28	24	21	331	13	22	27	12	340	15	21	26	13
323	18	22	22	28	332	18	19	24	19	341	21	17	25	19
324	19	23	23	31	333	20	22	24	24	342	25	22	21	29
325	23	23	23	36	334	25	23	25	32	343	28	23	25	33
326	26	22	17	43	335	21	30	22	36	344	22	31	13	44
327	14	41	17	49	336	22	38	22	45	345	15	46	13	50
328	40	31	2	58	337	13	55	-3	61	346	10	59	2	57
ctx_abs_level														
347	-9	55	-3	43	357	-11	63	-6	51	367	-11	63	-6	51
348	-1	30	-1	14	358	-3	34	-4	21	368	-3	34	-4	21
349	-2	34	0	16	359	-5	39	-6	28	369	-5	39	-6	28
350	-2	36	2	18	360	-5	41	-4	31	370	-5	41	-4	31
351	-1	37	1	23	361	-4	44	-5	37	371	-4	44	-5	37
352	-4	40	-7	36	362	-7	46	-10	41	372	-7	46	-10	41
353	-1	45	-2	43	363	-7	54	-9	49	373	-7	54	-9	49
354	-6	53	-4	47	364	-5	56	-7	51	374	-5	56	-7	51
355	-8	55	-5	49	365	-6	58	-8	53	375	-6	58	-8	53
356	-11	64	1	52	366	-11	69	-11	60	376	-11	69	-11	60

Annex A
Profile and level definitions

(This annex forms an integral part of this Recommendation | International Standard)

A.1 Introduction

Profiles and Levels ~~indicate~~ specify ~~to decoders~~ the capabilities needed to decode the coded data, and may be used to indicate interoperability points between individual decoder implementations.

NOTE - This Recommendation | International Standard does not include individually selectable "options" at the decoder, as this would increase interoperability difficulties.

Each Profile defines a set of algorithmic features and limits which shall be supported by all decoders conforming to that Profile. Note that encoders are not required to make use of any particular set of features supported in a Profile.

Each Level defines a set of limits on the values which may be taken by the parameters of this Recommendation International Standard. The same set of Level definitions is used with all Profiles, but individual implementations may support a different Level for each supported Profile. For any given Profile, Levels generally correspond to decoder processing and memory capability, in units based on video decoding, rather than on specific implementation platforms.

A.2 Requirements on video decoder capability

Capabilities of video decoders conforming to this Recommendation | International Standard are specified in terms of the ability to decode video streams conforming to constraints Profiles and Levels specified in this Annex. For each such Profile, the Level supported for that Profile shall also be expressed. Such expression may be in the form of coded values equivalent to a specific Profile and specific Level from this Annex.

Specific values are defined in this annex for the syntax elements profile_idc and level_idc. All other values of profile_idc and level_idc are reserved for future use by ITU-T | ISO/IEC.

NOTE: Decoders should not infer that if a reserved value of profile_idc or level_idc falls between the values specified in this Recommendation | International Standard this indicates intermediate capabilities between the specified profiles or levels, as there are no restrictions on the method to be chosen by ITU-T | ISO/IEC for the use of such future reserved values.

Formatted: Space Before: 6.8 pt

A.3 Baseline profile

A.3.1 Features

All decoders supporting the Baseline Profile shall be capable of decoding bitstreams which use the following features:

a) I and P picture types
b) In-loop deblocking filter
c) Frame pictures with mb_level_aff = 0
d) 1/4-sample motion compensation
e) Tree-structured motion segmentation down to 4x4 block size
f) VLC-based entropy coding
g) Arbitrary slice order (ASO): In Baseline profile, the decoding order of slices within a picture may not follow the constraint that first_mb_in_slice is monotonically increasing within the NAL unit stream for a picture.
h) Flexible macroblock ordering (FMO, maximum 8 slice groups): In Baseline profile, num_slice_groups_minus1<8.
i) Redundant slices
j) 4:2:0 Chrominance format

Decoders supporting the Baseline Profile Level 2.1 and above shall also be capable of decoding bitstreams using:

k) Field pictures.

Conformance to the Baseline Profile is indicated by setting the syntax element profile_idc equal to 66.

A.3.2 Limits

All decoders supporting this Profile shall be capable of decoding bitstreams which:

a) use 15 or fewer Reference Frames,
b) have a compression ratio per picture of 4:1 or greater,
c) use 64 or fewer Picture Parameter Sets, and,
d) use 16 or fewer Independent Sequence Parameter Sets.

A.4 X profile

A.4.1 Features

All decoders supporting the X Profile shall be capable of decoding bitstreams which use the following features:

Formatted: Space Before: 6.8 pt, Tab stops: 0.55", Left + 0.83", Left + 1.1", Left + 1.38", Left

a) Bi-predictive slices
b) SP and SI slices
c) Data partitioned slices
d) Weighted prediction
e) All features included in the Baseline Profile

All video decoders supporting the X Profile shall also support the Baseline Profile. The Level number supported for the Baseline Profile shall not be less than the Level number supported for the X Profile.

Conformance to the X Profile is indicated by setting the syntax element profile_idc equal to 88.

A.4.2 Limits

All decoders supporting this Profile shall be capable of decoding bitstreams which:
a) use 15 or fewer Reference Frames,
b) have a compression ratio per picture of 4:1 or greater,
c) use 64 or fewer Picture Parameter Sets, and,
d) use 16 or fewer Sequence Parameter Sets.

A.5 Main profile

A.5.1 Features

All decoders supporting the Main Profile shall be capable of decoding bitstreams which use the following features:

a) Bi-predictive slices
b) CABAC
c) Weighted prediction
d) Adaptive block-size transforms (ABT)
e) All features included in the Baseline Profile except:
   1. Arbitrary Slice Order (ASO): In Main profile, the decoding order of slices within a picture shall follow the constraint that first_mb_in_slice shall be monotonically increasing within the NAL unit stream for a picture.
   2. Flexible Macroblock Order (FMO): In Main profile, num_slice_groups_minus1 shall be zero.
   3. Redundant Slices

Decoders supporting Level 2.1 and above shall also be capable of decoding bitstreams using:

f) Interlaced pictures, frame/field adaptive at picture level and macroblock level.

Conformance to the Main profile is indicated by setting the syntax element profile_idc equal to 77.

A.5.2 Limits

All decoders supporting this Profile shall be capable of decoding bitstreams which:
a) use 15 or fewer Reference Frames,
b) have a compression ratio per picture of 4:1 or greater,
c) use 64 or fewer Picture Parameter Sets, and,
d) use 16 or fewer Sequence Parameter Sets.

A.6 Level definitions

A.6.1 General

Level limits are expressed in units of whole luma macroblocks. If a particular picture height or width is not an exact multiple of a whole macroblock, that dimension shall be considered as rounded up to the next whole macroblock for the purposes of conformance with this subclause.

The definition of support for a given Level is that any picture size/frame rate combination shall be decoded where the:

a) sample processing rate (in whole macroblocks/second) is <= the Level limit given, and,

b) picture size (Height * Width, in whole macroblocks) is <= the Level limit given, and,

c) required reference memory is <= the Level limit given, and,

d) maximum video bit rate of the bitstream is <= the Level limit given, and,

e) required HRD/VBV buffer size is <= the Level limit given, and,

f) horizontal and vertical motion vector range does not exceed the Level limits given, and,

g) picture Height and picture Width (in whole macroblocks) are <= sqrt(LevelLimitMaxPictureSize * 8), and,

h) frame rate is not greater than 172 Hz.

The definition of each Level includes the requirements of all lower numbered Levels in Table A-1. Decoders supporting a given Level shall also be capable of decoding bitstreams using all lower numbered Levels.

Note that display of decoded video is outside the scope of this Recommendation | International Standard; some decoder implementations may not include displays at all, and display limitations do not necessarily cause interoperability failures.

"Picture size" means the total number of macroblocks in the complete picture (both even and odd fields if interlaced).

A.6.2 Level limits

Table A-1 below gives the parameter limits for each Level. Conformance to a particular Level shall beis indicated by setting the syntax element level_idc equal to a value of ten times the level number specified in Table A-1.

Formatted: Note 1

Table A-1 – Level Limits

Level #	Max Sample Processing Rate (MB/s)	Max Picture Size (MBs)	Reference Memory (1024 bytes)	Max Video Bitrate (1000 bits/sec)	Max HRD/VBV Buffer Size (bits)	Horizontal MV Range (full pels)	Vertical MV Range (full pels)	Minimum luma Bi-predictive block size
1	1 485	99	148.5	64	163 840	[-2048, 2047.75]	[-64,+63.75]	8x8
1.1	2 970	396	891.0	128	327 680	[-2048, 2047.75]	[-128,+127.75]	8x8
1.2	5 940	396	891.0	768	1 966 080	[-2048, 2047.75]	[-128,+127.75]	8x8
2	11 880	396	891.0	2 000	2 000 000	[-2048, 2047.75]	[-128,+127.75]	8x8
2.1	19 800	792	1 782.0	4 000	4 000 000	[-2048, 2047.75]	[-256,+255.75]	8x8
2.2	20 250	1 620	3 037.5	4 000	4 000 000	[-2048, 2047.75]	[-256,+255.75]	8x8
3	40 500	1 620	3 037.5	8 000	8 000 000	[-2048, 2047.75]	[-256,+255.75]	8x8
3.1	108 000	3 600	6 750.0	20 000	20 000 000	[-2048, 2047.75]	[-512,+511.75]	8x8
3.2	216 000	5 120	7 680.0	20 000	20 000 000	[-2048, 2047.75]	[-512,+511.75]	8x8
4	245 760	8 192	12 288.0	20 000	20 000 000	[-2048, 2047.75]	[-512,+511.75]	8x8
5	491 520	19 200	28 800.0	TBD	(1s @ max bps)	[-2048, 2047.75]	TBD	8x8

Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Font: Not Bold
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next
Formatted: Keep with next

Levels with non-integer Level numbers in Table A-1 are refered to as "intermediate Levels". All Levels have the same status, but note that some applications may choose to use only the integer-numbered Levels.

Informative subclause A.7 shows the effect of these limits on frame rates for several example picture formats.

A.6.3 Reference memory constraints on modes

"Reference Memory" means the decoder memory pool which is used to store reference frames and post-decoder frame buffers. Note that P pictures require one reference frame, and B pictures require two reference frames; any remaining reference memory may be used either for multiple reference frames or for post-decoder frame buffers.

Decoders shall support all bitstreams within a given Profile and Level where the required reference memory does not exceed the limit given for the Level.

The reference memory required for a given mode shall be calculated as:

bytes = PictureSize * NumberOfReferenceFrames * ChromaFormatParameter * 256

The PictureSize parameter is in units of whole macroblocks.

The parameter ChromaFormatParameter shall take values according to Table A-2:

Table A-2 – ChromaFormatParameter values

Chrominance Format	ChromaFormatParameter
monochrome	1
4:2:0	1.5
4:2:2	2
4:4:4	3

A.7 Effect of level limits on frame rate (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Level number:					1	1.1	1.2	2	2.1	2.2	3	3.1	3.2	4	5
Max picture size (macroblocks):					99	396	396	396	792	1,620	1,620	3,600	5,120	8,192	19,200
Max macroblocks/second:					1,485	2,970	5,940	11,880	19,800	20,250	40,500	108,000	216,000	245,760	491,520
Max picture size (samples):					25,344	101,376	101,376	101,376	202,752	414,720	414,720	921,600	1,310,720	2,097,152	4,915,200
Max samples/second (1000s):					380	760	1,521	3,041	5,069	5,184	10,368	27,648	55,296	62,915	125,829
Format	Sample Width	Sample Height	MB Wide	MB High											
SQCIF	128	96	8	6	30.9	61.9	123.8	172.0	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QCIF	176	144	11	9	15.0	30.0	60.0	120.0	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QVGA	320	240	20	15	-	9.9	19.8	39.6	66.0	67.5	135.0	172.0	172.0	172.0	172.0
SIF	352	240	22	15	-	9.0	18.0	36.0	60.0	61.4	122.7	172.0	172.0	172.0	172.0
CIF	352	288	22	18	-	7.5	15.0	30.0	50.0	51.1	102.3	172.0	172.0	172.0	172.0
2SIF	352	480	22	30	-	-	-	-	30.0	30.7	61.4	163.6	172.0	172.0	172.0
HHR	352	576	22	36	-	-	-	-	25.0	25.6	51.1	136.4	172.0	172.0	172.0
VGA	640	480	40	30	-	-	-	-	-	16.9	33.8	90.0	172.0	172.0	172.0
4SIF	704	480	44	30	-	-	-	-	-	15.3	30.7	81.8	163.6	172.0	172.0
NTSC SD	720	480	45	30	-	-	-	-	-	15.0	30.0	80.0	160.0	172.0	172.0
4CIF	704	576	44	36	-	-	-	-	-	12.8	25.6	68.2	136.4	155.2	172.0
PAL SD	720	576	45	36	-	-	-	-	-	12.5	25.0	66.7	133.3	151.7	172.0
SVGA	800	600	50	38	-	-	-	-	-	-	-	56.8	113.7	129.3	172.0
XGA	1024	768	64	48	-	-	-	-	-	-	-	35.2	70.3	80.0	160.0
720p	1280	720	80	45	-	-	-	-	-	-	-	30.0	60.0	68.3	136.5
4VGA	1280	960	80	60	-	-	-	-	-	-	-	-	45.0	51.2	102.4
SXGA	1280	1024	80	64	-	-	-	-	-	-	-	-	42.2	48.0	96.0
16SIF	1408	960	88	60	-	-	-	-	-	-	-	-	-	46.5	93.1
16CIF	1408	1152	88	72	-	-	-	-	-	-	-	-	-	38.8	77.6
4SVGA	1600	1200	100	75	-	-	-	-	-	-	-	-	-	32.8	65.5
1080i	1920	1080	120	68	-	-	-	-	-	-	-	-	-	30.1	60.2
2Kx1K	2048	1024	128	64	-	-	-	-	-	-	-	-	-	30.0	60.0
4XGA	2048	1536	128	96	-	-	-	-	-	-	-	-	-	-	40.0
16VGA	2560	1920	160	120	-	-	-	-	-	-	-	-	-	-	25.6

Note 1 This is a variable-picture-size specification. The specific picture sizes in this table are illustrative examples only.

Note 2 XGA is also known as (aka) XVGA, 4SVGA aka UXGA, 16XGA aka 4Kx3K, HHR aka 2CIF aka 1/2 D1, aka 1/2 ITU-R BT.601.

Note 3 Frame rates given are correct for progressive scan modes, and for interlaced if "MB High" column value is even.

Annex B

Byte stream format

(This annex forms an integral part of this Recommendation | International Standard)

B.1 Introduction

This annex defines a byte stream format specified for use by systems that transmit some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as ITU-T Recommendation H.222.0 | ISO/IEC 13818-1 systems or ITU-T Recommendation H.320 systems. For bit-oriented transmission, the network bit order for the byte stream format is defined to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

The byte stream format consists of a sequence of byte_stream_unit( ) structures. Each byte_stream_unit( ) contains one start code prefix (SCP) and one nal_unit( ). Optionally, at the discretion of the encoder, the byte_stream_unit( ) may also contain additional "stuffing" zero-valued bytes as specified in this clause.

There are two types of start code prefixes:

- A short SCP, consisting of one byte having the value zero (0x00) followed by one byte having the value one (0x01), and
- A long SCP, consisting of two bytes having the value zero (0x00) followed by one byte having the value one (0x01).

The long SCP provides a mechanism for decoder byte-alignment recovery in the event of loss of decoder synchronization.

B.2 Byte stream NAL unit syntax

byte_stream_unit() {	**Category**	**Mnemonic**
while (next_bits(16) != 0x0001 && next_bits(24) != 0x000001)		
zero_byte		f(8) = 0x00
if(next_bits() – – 0x000001)		
zero_byte		f(8) = 0x00
zero_byte		f(8) = 0x00
one_byte		f(8) = 0x01
nal_unit( )		
}		

B.3 Byte stream NAL unit semantics

The order of byte stream NAL units in the byte stream shall follow the decoding order of the NAL units contained in the byte stream NAL units.

zero_byte is a single byte (8 bits) having the value zero (0x00). Optionally, at the discretion of the encoder, the beginning of a byte_stream_unit() may contain more zero_byte syntax elements than required in this subclause.

The minimum required number of zero_byte syntax elements depends on the nal_unit_type as defined in Table 7-1, in order to ensure the use of the long SCP for certain nal_unit_type values. This ensures use of the long SCP in the byte_stream_unit() for these nal_unit() structures. The use of the long SCP for nal_unit() structures with other values of nal_unit_type is optional. At least two zero_byte syntax elements shall be present in each byte_stream_unit() for the following values of nal_unit_type:

- 0x06: Supplemental enhancement information,
- 0x07: Sequence parameter set,
- 0x08: Picture parameter set, and
- 0x09: Picture delimiter.

one_byte is a single byte (8 bits) having the value one (0x01). A sequence of two zero_byte syntax elements followed by a one_byte is a long SCP, and one zero_byte followed by a one_byte is a short SCP.

The use of the byte sequence 0x00 0x02 is reserved for future use by ITU-T | ISO/IEC.

B.4 Decoder byte-alignment recovery (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Formatted: Don't keep with next

If the decoder does not have byte alignment with the encoder's byte stream, the decoder can examine the incoming bit stream for the binary pattern '00000000 00000000 00000001' (23 consecutive zero-valued bits followed by a non-zero bit). The bit immediately following this pattern is the first bit of a whole byte. Upon detecting this pattern, the decoder will be byte aligned with the encoder.

Once byte aligned with the encoder, the decoder can examine the incoming byte stream for the byte sequences 0x00 0x01 and 0x00 0x03.

If the byte sequence 0x00 0x01 is detected, this represents a SCP. If the previous byte was 0x00, the SCP is a long SCP. Otherwise, it is a short SCP.

If the byte sequence 0x00 0x03 is detected, the decoder discards the byte 0x03 as shown in the rbsp_extraction() syntax diagram.

NOTE - Many systems are inherently byte aligned, and thus have no need for the bit-oriented byte alignment detection procedure described in this subclause.

NOTE - The byte alignment detection procedure described in this subclause is equivalent to searching a byte sequence for 0x00 0x00, starting at any alignment position. Detecting this pattern indicates that the next non-zero byte contains the end of a SCP, and the first non-zero bit in that next non-zero byte is the last bit of an aligned byte.

Annex C
Hypothetical Reference Decoder

(This annex forms an integral part of this Recommendation | International Standard)

C.1 Hypothetical reference decoder and buffering verifiers

The hypothetical reference decoder (HRD) represents a set of normative requirements on coded bitstreams or packet streams. These constraints must be enforced by an encoder, and can be assumed by a decoder or multiplexor to be true. It is possible to verify the conformance of a bitstream or packet stream to the requirements of this subclause by examining the bitstream or packet stream only.

This subclause defines the normative requirements of the HRD. Subclause C.2 provides additional information that is important for a full understanding of the HRD operation.

Two types of streams may be subject to the HRD requirements of this Recommendation | International Standard; a stream of VCL NAL Units and a bitstream. Figure C-1 shows how each of these is constructed from the RBSP. In other words, a given set of HRD parameters may pertain to the VCL data only or to the multiplexed combination of VCL and NAL. This is signalled through video usability information (subclauses E.2 and E.3).

.

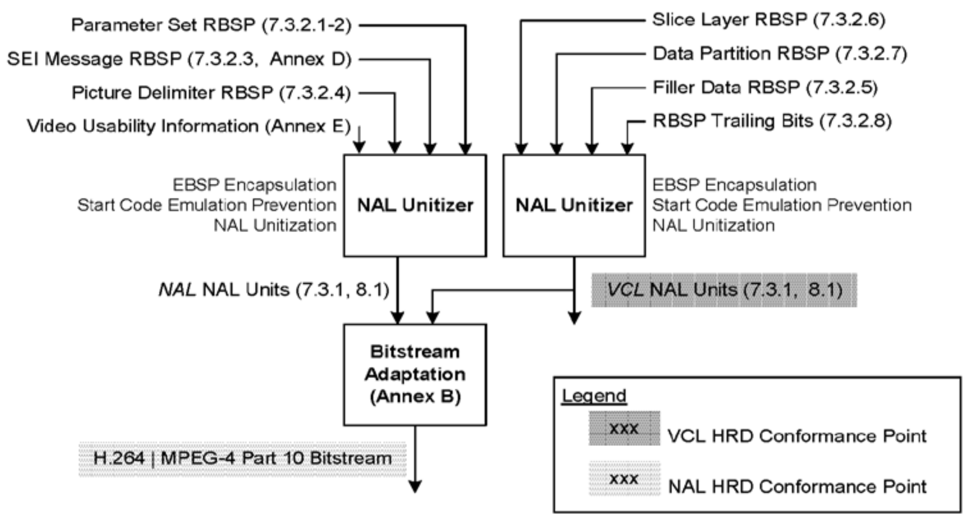

Figure C-1 – Structure of Byte streams and NAL unit streams and HRD Conformance Points

The HRD can contain any combination of the following buffering verifiers, as shown in Figure C-2:

- One or more pre-decoder buffers, each of which is either variable bit rate (VBR) or constant bit rate (CBR)
- At most one reference and post-decoder buffer attached to the output of one of the pre-decoder buffers

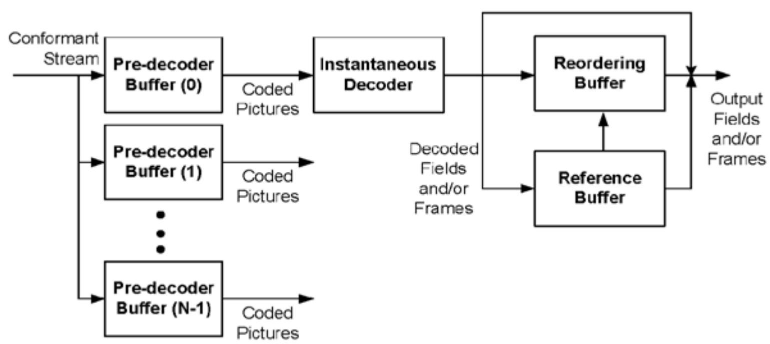

Figure C-2 – HRD Buffer Verifiers

The multiple buffering verifiers exist because a bit stream or packet stream may conform to multiple pre-decoder buffers, as detailed in subclause C.2.2.

All the arithmetic in this annex is done with real values, so that no rounding errors can propagate. For example, the number of bits in a pre-decoder buffer just prior to or after removal of a transmitted picture is not necessarily an integer. Furthermore, while conformance is guaranteed under the assumption that all frame-rates and clocks used to generate the bitstream match exactly the values signalled in the bitstream, each of these may vary from the signalled or defined value.

This hypothetical reference decoder uses two time bases. One time base is a 90 kHz clock, and is only in operation for a short time after the reception of a Buffering Period SEI message. The second time base uses the num units in tick and time_scale syntax in the Sequence Parameter Set to derive the time interval between picture removals from the buffers (and in some cases between picture arrivals to the pre-decoder buffer).

In the following description, let t_c = num_units_in_tick ÷ time_scale be the *clock tick* associated with the second clock. The clock tick is a time interval no larger than the shortest possible inter-picture capture interval in seconds. Also let $be[t]$ and $te[b]$ be the bit equivalent of a time t and the time equivalent of a number of bits b, with the conversion factor being the buffer arrival bit rate.

The following statements are normative requirements on the composition of a conforming bitstream. If multiple sequence parameter sets pertain to the bit stream or packet stream, they must contain consistent HRD information. In the case that any HRD buffers are signalled in the sequence parameter set(s), then the following rules dictate the insertion of SEI messages in the bit stream or packet stream.

1. At each decoder refresh point (IDR, ODR or GDR), a buffering period SEI message shall follow the last NAL Unit of the last picture before a decoder refresh and precede the first NAL Unit of the first picture after the decoder refresh. Note that in the case of an IDR, this SEI message will precede the indication of the decoder refresh point.
2. An HRD picture SEI message must follow the last NAL Unit of each picture and precede the first NAL Unit of the next picture. Each of these SEI messages pertains to the picture that follows it.

C.1.1 Operation of VCL video buffering verifier (VBV) pre-decoder buffer

This specification applies independently to each pre-decoder buffer VUI sequence parameters within the sequence parameter set.

C.1.1.1 Timing of bitstream or packet stream arrival

The buffer is initially empty. The first bit of the first transmitted picture begins to enter the buffer at *initial arrival time* $t_{ai}(0)=0$ at the bit rate bit_rate[k] associated with the pre-decoder buffer (see subclause 8.3.3). The last bit of the first transmitted picture finishes arriving at *final arrival time*

$$t_{af}(0) = b(0) \div \text{bit\_rate[k]}, \qquad \text{(C-1)}$$

where $b(n)$ is the size in bits of the n-th transmitted picture. The final arrival time for each picture is always the sum of the initial arrival time and the time required for the bits associated with that picture to enter the pre-decoder buffer:

$$t_{af}(\mathrm{n}) = t_{ai}(\mathrm{n}) - b(\mathrm{n}) \div \text{bit\_rate[k]}. \qquad \text{(C-2)}$$

For each subsequent picture, the initial arrival time of picture n is the later of $t_{af}(n\text{-}1)$ and the sum of all preceding **pre_dec_removal_delay** times, as indicated in Equation C-3.

$$t_{ai}(n) = \max\{\ t_{af}(n\text{-}1),\ t_c \times \sum_{m=0}^{n-1} \text{pre\_dec\_removal\_delay}(m)\ \} \qquad \text{(C-3)}$$

See subclauses C.2.5 and C.3.5 for the syntax and semantics of the pre_dec_removal_delay times. When the encoder is producing a bit rate lower than the bit rate associated with a pre-decoder buffer, this rule may delay the entry of some pictures into the pre-decoder buffer, producing periods during which no data enters.

C.1.1.2 Timing of coded picture removal

For the first picture and all pictures that are the first complete picture after receiving a buffering period SEI message, the coded data associated with the picture is removed from the pre-decoder buffer at a *removal time* equal to the following:

$$t_r(0) = \text{initial\_pre\_dec\_removal\_delay} \div 90000 \qquad \text{(C-4)}$$

where initial_pre_dec_removal_delay is the pre-decoder removal delay in the buffering period SEI message.

After the first picture is removed, the buffer is examined at subsequent points of time, each of which is delayed from the previous one by an integer multiple of the clock tick t_c.

The removal time $t_r(n)$ of coded data for picture n is delayed with respect to that of picture $n\text{-}1$; the delay is equal to the time indicated in the pre_dec_removal_delay syntax element present in the HRD picture SEI message.

$$t_r(n) = t_r(n-1) + t_c \times \text{pre\_dec\_removal\_delay}(n) \quad \text{(C-5)}$$

At this time, the coded data for the next transmitted picture is removed from the pre-decoder buffer.

In the case that the amount of coded data for picture n, $b(n)$, is so large that it prevents removal at the computed removal time, the coded data is removed at the *delayed removal time*, $t_{r,ld}(n, m^*)$, given by

$$t_{r,d}(n,m^*) = t_r(0) + t_c \times m^*, \quad \text{(C-6)}$$

where m^* is such that $t_{r,d}(n, m^*-1) < t_{af}(n) \le t_{r,ld}(n, m^*)$. This is an aspect of low-delay operation (see subclause C.2.1.2). This delayed removal time is the next time instant after the final arrival time $t_{af}(n)$ which is delayed with respect to $t_r(0)$ by an integer multiple of t_c.

C.1.1.3 Conformance constraints on coded bitstreams or packet streams

A transmitted or stored stream of coded data conforming to this Recommendation | International Standard fulfils the following requirements.

- ***Removal time consistency.*** For each picture, the removal times $t_r(n)$ computed using different buffering periods as starting points for conformance verification shall be consistent to within the accuracy of the two clocks used (90 kHz clock used for initial removal time and t_c clock used for subsequent removal time calculations). This can be ensured at the encoder by computing the pre-decoder removal delay (initial_pre_dec_removal_delay) for a buffering period SEI message from the arrival and removal times computed using Equations C-3 and C-5. Any small deviations between the values computed in the different ways shall not cause violation of any of the following constraints.

- ***Underflow and Overflow Prevention.*** The buffer must never overflow or underflow.

NOTE - In terms of the arrival and removal schedules, this means that, with the exception of some pictures in low-delay mode that are described below, all bits from a picture must be in the pre-decoder buffer at the picture's computed removal time tr(n). In other words, its final arrival time must be no later than its removal time: taf(n) ≤ tr(n). Further, the removal time tr(n) must be no later than the time-equivalent of the buffer size te[pre_dec_buffer_size[k]]. Note that this prevents overflow.

- ***Big Picture Removal Time, Overflow Prevention and Resynchronisation of Underflow Prevention.*** If the final arrival time $t_{af}(n)$ of picture n exceeds its computed removal time $t_r(n)$, its size must be such that it can be removed from the buffer without overflow at $t_{r,d}(n,m^*)$ as defined above.

- ***Constant Bit Rate Constraint.*** If **vbr_cbr_flag[k]** = = 1, data shall arrive continuously at the input to the pre-decoder buffer. This is equivalent to ensuring that $t_{af}(n-1) \ge t_c \times \sum_{m=0}^{n-1} \text{pre\_dec\_removal\_delay}(m)$.

- ***Time Duration Constraint.*** For each picture immediately preceding a Buffering Period SEI message, the sum of pre-decoder removal delays from the start of the sequence up to that point of time shall be no further from the accumulated sequence duration as represented by the sum of prev_buf_period_duration than the removal_time_tolerance in the relevant sequence parameter set.

  If the picture immediately preceding the Buffering Period SEI message is the n-th picture in transmitted order, and the buffering period SEI message is the k-th such message, then this constraint amounts to the following:

$$\left| \sum_{m=0}^{n-1} \text{pre\_dec\_removal\_delay(m)} - \sum_{m=0}^{k-1} \text{prev\_buf\_period\_duration(m)} \right| \le \text{removal\_time\_tolerance} \quad \text{(C-7)}$$

- ***Maximum Decoder Frame Rate.*** The interval between consecutive removal times shall not be lower than the minimum picture interval, defined as the inverse of the maximum picture rate (See A.5.1).

C.1.2 Operation of the post-decoder buffer verifier

C.1.2.1 Arrival timing

A reconstructed picture is added to the post-decoder buffer at the same time when the corresponding coded picture is removed from the pre-decoder buffer.

C.1.2.2 Removal timing

Data is not removed from the post-decoder buffer during a period called the initial post-decoder buffering period. The period starts when the first picture is added to the post-decoder buffer.

When the initial post-decoder buffering period has expired, the playback timer is started from the earliest display time of the pictures residing in the post-decoder buffer at that time.

A picture is virtually displayed when the playback timer reaches the scheduled presentation time of the picture.

A picture memory is marked unused in the post-decoder buffer when it is virtually displayed and when it is no longer needed as a reference picture.

C.1.2.3 Conformance constraints

The occupancy of the post-decoder buffer shall not exceed the default or signalled buffer size.

Each picture shall be available in the post-decoder buffer before or on its presentation time.

C.2 Informative description of the HRD

Subclause C.1 contains the normative requirements imposed by a set of buffering verifiers. This subclause provides explanatory text describing in more detail the operation and capabilities of these buffers.

An HRD represents a means to communicate how the bit rate is controlled in the process of compression. The HRD contains a pre-decoder buffer (or VBV Buffer) through which compressed data flows with a precisely specified arrival and removal timing, as shown in Figure C-3. An HRD may be designed for variable or constant bit rate operation, and for low-delay or delay-tolerant behavior. The HRD described in this document handles all cases.

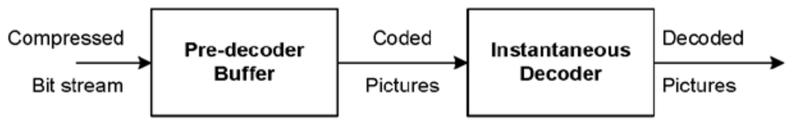

Figure C-3 – A Hypothetical Reference Decoder

Compressed data representing a sequence of *coded pictures* flows into the pre-decoder buffer according to a specified *arrival schedule*. All compressed bits associated with a given coded picture are removed from the pre-decoder buffer by the instantaneous decoder at the specified *removal time* of the picture.

The pre-decoder buffer *overflows* if the buffer becomes full and more bits are arriving. The buffer *underflows* if the removal time for a picture occurs before all compressed bits representing the picture have arrived.

C.2.1 Constrained arrival time leaky bucket (CAT-LB) model

The hypothetical reference decoder (HRD) is a mathematical model of a decoder and its input buffer. The k-th pre-decoder buffer of the HRD is characterized by the pre-decoder peak rate bit_rate[k] (in bits per second), the buffer size pre_dec_buffer_size[k] (in bits), the sequence initial pre-decoder buffer removal delay (in seconds), as well as picture removal delays for each picture. The first three of these parameters represent levels of resources (transmission capacity, buffer capacity, and delay) used to decode a bitstream.

The term "leaky bucket" arises from the analogy of the encoder as a system that "dumps" water in discrete chunks into a bucket that has a hole in it. The departure of bits from the encoder buffer corresponds to water leaking out of the bucket. Here, the decoder buffer is described, which has an inverse behaviour where bits flow *in* at a constant rate, and are removed in chunks.

The leaky bucket described here is called a *constrained arrival time* leaky bucket because the arrival times of all pictures after the first are constrained to arrive at the buffer input no earlier than the difference in hypothetical encoder processing times between that picture and the first picture. In other words, if a picture is encoded exactly seven seconds after the first picture was encoded, then its bits are guaranteed not to start arriving in the buffer prior to seven seconds after the bits of the first picture started arriving. It is possible to know this encoding time difference because it is sent in the bitstream as the picture removal delay.

C.2.1.1 Operation of the CAT-LB HRD

The HRD input buffer has capacity pre_dec_buffer_size[k] bits. Initially, the buffer begins empty. The lifetime in the buffer of the coded bits associated with picture n is characterized by the *arrival interval* $\{t_{ai}(n), t_{af}(n)\}$ and the *removal time* $t_r(n)$. The end-points of the arrival interval are known as the *initial arrival time* and the *final arrival time*.

At time $t_{ai}(0) = 0$, the buffer begins to receive bits at the rate bit_rate[k]. The removal time $t_r(0)$ for the first picture is computed from the pre-decoder removal delay initial_pre_dec_removal_delay (see Buffering Period SEI Message) associated with the buffer by the following:

$$t_r(0) = 90{,}000 \times \text{initial\_pre\_dec\_removal\_delay}. \qquad \text{(C-8)}$$

Removal times $t_r(1)$, $t_r(2)$, $t_r(3)$, …, for subsequent pictures (in transmitted order) are computed with respect to $t_r(0)$, as follows. Let the *clock tick* t_c be defined by

$$t_c = \text{num\_units\_in\_tick} \div \text{time\_scale} \qquad \text{(C-9)}$$

For instance, if time_scale = 60,000 and num_units_in_tick = 1,001, then

$$t_c = 1{,}001 \div 60{,}000 = 16.68333\ldots \text{ milliseconds.} \qquad \text{(C-10)}$$

In the HRD picture SEI message for each picture, there is a pre_dec_removal_delay syntax element. This indicates the number of clock ticks to delay the removal of picture n after removing picture $n\text{-}1$. Thus, the removal time is simply

$$t_r(n) = t_r(n\text{-}1) + t_c \times \text{pre\_dec\_removal\_delay}(n)\text{(C-11)}$$

Note that this recursion can be used to show that

$$t_r(n) = t_r(0) + t_c \times \sum_{m=1}^{n} [\text{pre\_dec\_removal\_delay}(m)], \qquad \text{(C-12)}$$

The calculation of arrival times is more complex, because of the arrival time constraint. The initial arrival time of picture n is equal to the final arrival time of picture $n\text{-}1$, unless that time precedes the earliest arrival time, computed by

$$t_{ai,earliest}(n) = t_c \times \sum_{m=1}^{n} [\text{pre\_dec\_removal\_delay}(m)] \qquad \text{(C-13)}$$

Let $b(n)$ be the number of bits associated with picture n. The duration of the picture arrival interval is always the time-equivalent of the picture size in bits, at the rate bit_rate[k].

$$t_{af}(n) - t_{ai}(n) \equiv te[b(n)] = b(n) \div \text{bit\_rate[k]} \qquad \text{(C-14)}$$

Figure C-4 demonstrates a segment of the pre-decoder buffer fullness plot for a CAT-LB with the parameters given in Table C-1 and picture sizes given by the first column of Table C-2. Note that Table C-2 lists for each picture the values for many times of interest in the buffering process. In addition to quantities defined above, the second column of Table C-2 contains t_e, which represents a hypothetical encoding time equal to the earliest possible initial arrival time of the picture.

Table C-1 - Attributes of an example CAT-LB HRD

Attribute	Value	Units
time_scale	1	units per second
num_units_in_tick	1	units per tick
bit_rate	1000	bits per second
pre_dec_buffer_size	10	bits
initial_delay	10	seconds

Table C-2 - Picture sizes, and encoding, arrival and removal times for the example CAT-LB HRD

b	t_e	t_{ai}	t_{af}	t_{ai}-t_e	t_r	t_r - t_{ai}	t_r - t_e
5,000	0	0	5	0	10	10	10
1,000	1	5	6	4	11	6	10
1,000	2	6	7	4	12	6	10
1,000	3	7	8	4	13	6	10
1,000	4	8	9	4	14	6	10
1,000	5	9	10	4	15	6	10
500	6	10	10.5	4	16	6	10
500	7	10.5	11	3.5	17	6.5	10
500	8	11	11.5	3	18	7	10
500	9	11.5	12	2.5	19	7.5	10
500	10	12	12.5	2	20	8	10
500	11	12.5	13	1.5	21	8.5	10
500	12	13	13.5	1	22	9	10
500	13	13.5	14	0.5	23	9.5	10
500	14	14	14.5	0	24	10	10
500	15	15	15.5	0	25	10	10
500	16	16	16.5	0	26	10	10
500	17	17	17.5	0	27	10	10
3,000	18	18	21	0	28	10	10
3,000	19	21	24	2	29	8	10
3,000	20	24	27	4	30	6	10
3,000	21	27	30	6	31	4	10
2,000	22	30	32	8	32	2	10
300	23	32	32.3	9	33	1	10
300	24	32.3	32.6	8.3	34	1.7	10
300	25	32.6	32.9	7.6	35	2.4	10
300	26	32.9	33.2	6.9	36	3.1	10
300	27	33.2	33.5	6.2	37	3.8	10
300	28	33.5	33.8	5.5	38	4.5	10
300	29	33.8	34.1	4.8	39	5.2	10
300	30	34.1	34.4	4.1	40	5.9	10
300	31	34.4	34.7	3.4	41	6.6	10
300	32	34.7	35	2.7	42	7.3	10
300	33	35	35.3	2	43	8	10
300	34	35.3	35.6	1.3	44	8.7	10
300	35	35.6	35.9	0.6	45	9.4	10
300	36	36	36.3	0	46	10	10
300	37	37	37.3	0	47	10	10
300	38	38	38.3	0	48	10	10
300	39	39	39.3	0	49	10	10
300	40	40	40.3	0	50	10	10
300	41	41	41.3	0	51	10	10
300	42	42	42.3	0	52	10	10
500	43	43	43.5	0	53	10	10
500	44	44	44.5	0	54	10	10

500	45	45	45.5	0	55	10	10
500	46	46	46.5	0	56	10	10
500	47	47	47.5	0	57	10	10
500	48	48	48.5	0	58	10	10
500	49	49	49.5	0	59	10	10
500	50	50	50.5	0	60	10	10
500	51	51	51.5	0	61	10	10
500	52	52	52.5	0	62	10	10

Legend:

t_e Encoding time

t_{ai} Initial arrival time

t_{af} Final arrival time

t_r Removal time

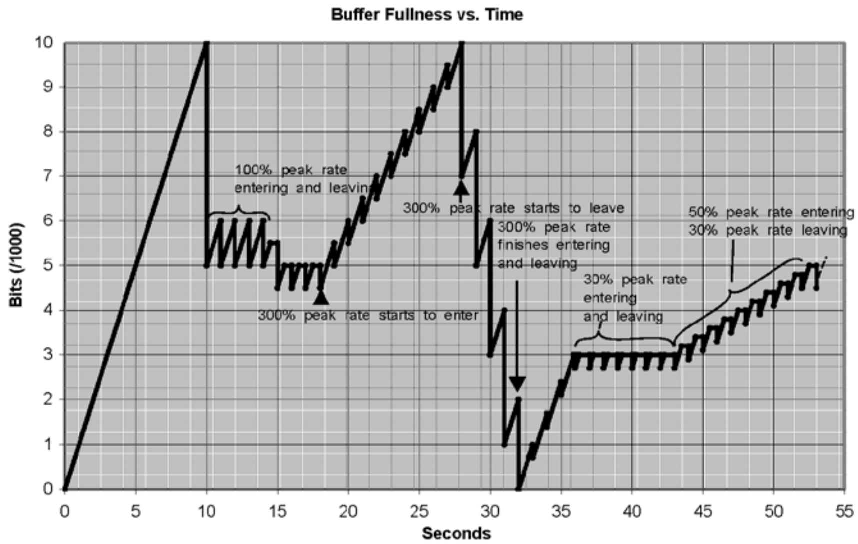

Figure C-4 – Buffer fullness plot for example HRD in Table C-2 with picture sizes given in Table C-3

As can be seen from Table C-2, the initial picture is large, and is followed by five pictures at exactly the buffer arrival rate R. This is followed by twelve pictures at half the rate, four pictures at three times the rate and one picture at twice the rate. Following this are two segments with pictures at 30% and 50% of the rate, respectively. In Figure C-4, the time interval from 10 seconds to 18 seconds illustrates the behaviour when the bit rate is constant and at or below the rate R. In fact, whenever the arrival bit rate remains less than R for a time, the lower points of the fullness curve will not change. Further, the fullness at the peak in such a segment will be proportional to the fraction of the peak rate being consumed by the pictures. From seconds 18 to 28, we see the temporary effect of an increase in arrival rate to above R. Once those large pictures start to exit the buffer, the bit rate of pictures leaving the buffer exceeds R, and the fullness decreases. This process terminates at second 32, when the big pictures have exited and the series of smaller pictures starts entering the buffer. During seconds 36-43, the 30% peak rate pictures are entering and leaving the buffer, and during seconds 43-52, 30% peak rate pictures are leaving while 50% peal rate pictures are entering. Hence the buffer fullness rises. Once 50% peak rate pictures begin to leave, the fullness stabilizes at 50% full. Note that this pre-decoder buffer stabilizes at a fullness that is proportional to the ratio of the short-term average bit rate to the arrival bit rate, rather than at 100%..

In general, the curve of buffer fullness vs. time is given by the following expression:

$$BF(t) = \sum_{n} [I(t_{af}(n) \le t < t_r(n)) \times b(n) + I(t_{ai}(n) < t < t_{af}(n)) \times be(t\text{-}t_{ai}(n))] \quad \text{(C-15)}$$

This expression uses indicator functions $I(\cdot)$ with time-related logical assertions as arguments to sum only those pictures that are completely in the buffer at time t, plus the appropriate portion of the picture currently entering the buffer, if one is. The indicator function $I(x)$ is '1' if x is true and '0' otherwise.

C.2.1.2 Low-delay operation

Low-delay operation is obtained by selecting a low value for the initial pre-decoder removal delay. This results in true low delay through the buffer because, under normal operation, no removal delay ($t_r(n)$-$t_{ai}(n)$) can exceed the initial removal delay $t_r(0)$. To see this, consider that the maximum removal delay for picture n occurs when the initial arrival time is equal to the earliest arrival time. Therefore, the maximum removal delay is given by $t_r(n)$ - $t_{ai,earliest}(n)$. But,

$$t_r(n) = t_r(0) + t_c \times \sum_{m=1}^{n} [\text{pre\_dec\_removal\_delay(m)}],$$

and

$$t_{ai,earliest}(n) = t_c \times \sum_{m=1}^{n} [\text{pre\_dec\_removal\_delay(m)}],$$

so

$$t_r(n) - t_{ai,earliest}(n) = t_r(0). \quad \text{(C-16)}$$

Thus setting an initial low delay creates a steady-state low-delay condition.

However, in low-delay operation, it is useful to be able to process the occasional large picture whose size is so large than that it cannot be removed by its indicated removal time. Such a large picture can arise at a scene change, for example. This would ordinarily lead to an "underflow" condition. When a large picture is encountered, the rules for removal are relaxed to prevent this. The picture is removed at the *delayed removal time*, $t_{r,ld}(n, m^*)$, given by

$$t_{r,ld}(n,m^*) = t_r(0) + t_c \times m^*, \quad \text{(C-17)}$$

where m^* is such that $t_{r,ld}(n, m^*\text{-}1) < t_{ai}(n) + \text{te}[b(n)] \le t_{r,ld}(n, m^*)$. Note that the buffer must be large enough that this large picture can be accommodated without overflow. Immediately after such a picture is received the removal time of the next picture must be such that low-delay operation is resumed. An encoder can facilitate this by skipping a number of pictures immediately after the large picture, if necessary.

C.2.1.3 Bitstream / packet stream constraints

The buffer must not be allowed to underflow or overflow. Furthermore, all pictures except the isolated big pictures must be completely in the buffer before their computed removal times. Isolated big pictures are allowed to arrive later than their computed removal times, but must still obey the overflow constraint. In CBR mode, there must be no gaps in bit arrival.

C.2.1.3.1 Underflow

The underflow constraint, $BF(t) \ge 0$ for all t, is satisfied if the final arrival time of each picture precedes its removal time.

$$t_{af}(n) \le t_r(n) \quad \text{(C-18)}$$

This puts an upper bound on the size of picture n. The picture size can be no larger than the bit-equivalent of the time interval from the start of arrival to the removal time.

$$b(n) \le be[t_r(n) - t_{ai}(n)] \quad \text{(C-19)}$$

Since the initial arrival time $t_{ai}(n)$ is in general a function of the sizes and removal delays of previous pictures, the constraint on $b(n)$ will vary over time as well.

C.2.1.3.2 Overflow

Overflow is avoided provided the buffer fullness curve $BF(t)$ never exceeds the buffer size B.

The constraints that the initial pre-decoder removal delay must be no larger than the time-equivalent of the buffer size, $t_r(0) \le te(B)$, and that under normal operation no removal delay can exceed the initial one guarantee that no overflow occurs in normal operation. To avoid overflow of an isolated big picture, the picture size is constrained by

$$b(n) \le be[B - t_{ai}(n)] \quad \text{(C-20)}$$

C.2.1.3.3 Constant bitrate (CBR) operation

The CAT-LB model operates in constant bit rate mode if one further constraint is applied - that data must constantly arrive at the input of the buffer. This ensures that the average rate is equal to the buffer rate R. This model behaves like an MPEG-1 CBR model with variable frame rate. This condition is ensured if the final arrival time of picture n is no earlier than the earliest initial arrival time of picture $n+1$.

$$t_{af}(n) \ge t_{ai,earliest}(n+1) = t_c \times \sum_{m=1}^{n} [\text{pre\_dec\_removal\_delay(m)}] \quad \text{(C-21)}$$

This time constraint puts a lower bound on $b(n)$.

C.2.1.4 Rate control considerations

An encoder employs rate control as a means to constrain the varying bit rate characteristics of the coded bitstream or packet stream in order to produce high quality coded pictures at the target bit rate(s). A rate control algorithm may target a variable bit rate (VBR) or a constant bit rate (CBR). It may even target both a high peak rate using a VBR scheme and an average rate using a CBR scheme. Further, as shown in subclause C.2.2, multiple VBR rates can be targeted.

Rate control must ensure conformance with the pre-decoder buffers. This is related to the first goal of rate control, but is not necessarily the same. In this subclause, the way the pre-decoder buffers influences rate control is discussed. In a VBR pre-decoder buffers, the buffer must not overflow or underflow, but gaps may appear in the arrival rate. In order to meet these constraints, the encoder must ensure that for all t, the following inequalities remain true:

$$0 \le BF(t) \le B, \text{ for all } t. \quad \text{(C-22)}$$

Using Equation D-14, this becomes:

$$0 \le \sum_{n} [I(t_{af}(n) \le t < t_r(n)) \times b(n) + I(t_{ai}(n) < t < t_{af}(n)) \times be(t - t_{ai}(n))] \le B, \text{ for all } t. \quad \text{(C-23)}$$

The buffer fullness $B(t)$ is a piecewise non-decreasing function of time, with each non-decreasing interval bounded by two consecutive removal times. Therefore, it is sufficient to guarantee the conformance at the interval endpoints; *i.e.* at the removal times. In particular, if underflow is prevented at the start of an interval (just after removal of a picture), it is completely prevented. The same holds for overflow at the end of the interval, just prior to picture removal. Therefore, the points of interest are the removal times. In the buffer at $t_r^-(n)$ and contributing to Equation C.22 are picture n and possibly some additional pictures up to picture $m>n$ (with the last picture possibly only partially in the buffer). All pictures earlier than picture n have been removed. At $t_r^+(n)$, picture n has been removed.

Thus when encoding picture n, one rate control task is to allocate bits to picture n and the others in the immediate future in such a way that overflow is prevented at $t_r^-(n)$, and underflow is prevented at $t_r^+(n)$. Most immediately, as long as $b(n)$ is small enough so that $te[b(n)] \le t_r(n) - t_{ai}(n)$, both overflow at $t_r^-(n)$ and underflow at $t_r^+(n)$ are prevented. This is usually a very high limit, and a rate control method will most likely further limit $b(n)$ through its bit allocation process.

C.2.2 Multiple leaky bucket description

C.2.2.1 Schedule of a bitstream

The sequence of removal time and picture size pairs $\{(t_r(n), b(n)), n=0,1,\ldots\}$ is called the schedule of a bitstream. The schedule of a bitstream is intrinsic to the bitstream, and completely characterizes the instantaneous coding rate of the bitstream over its lifetime. Although a bitstream may conform to VBVs with different peak bit rates and different pre-decoded buffer sizes, the schedule of the bitstream is independent of the VBV.

C.2.2.2 Containment in a leaky bucket

A leaky bucket with leak rate R_1, bucket size B_1, and initial bucket fullness $B_1–F_1$ is said to contain a bitstream with schedule $\{(t_i(n), b(n)), n=0,1,\ldots\}$ if the bucket does not overflow under the following conditions. At t_0, d_0 bits are inserted into the leaky bucket on top of the $B_1–F_1$ bits already in the bucket, and the bucket begins to drain at rate R1 bits per second. If the bucket empties, it remains empty until the next insertion. At time t_i, $i \geq 1$, d_i bits are inserted into the bucket, and the bucket continues to drain at rate R_1 bits per second. In other words, for $i \geq 0$, the state of the bucket just prior to time t_i is

$$b_0 = B_1–F_1 \tag{C-24}$$

$$b_{i+1} = \max\{0, b_i + d_i – R_1(t_{i+1}–t_i)\}. \tag{C-25}$$

The leaky bucket does not overflow if $b_i + d_i \leq B_1$ for all $i \geq 0$.

Equivalently, the leaky bucket contains the bitstream if the graph of the schedule of the bitstream lies between two parallel lines with slope R_1, separated vertically by B_1 bits, possibly sheared horizontally, such that the upper line begins at F_1 at time t_0, as illustrated in Figure C-5. Note from Figure C-5 that the same bitstream is containable in more than one leaky bucket. Indeed, a bitstream is containable in an infinite number of leaky buckets.

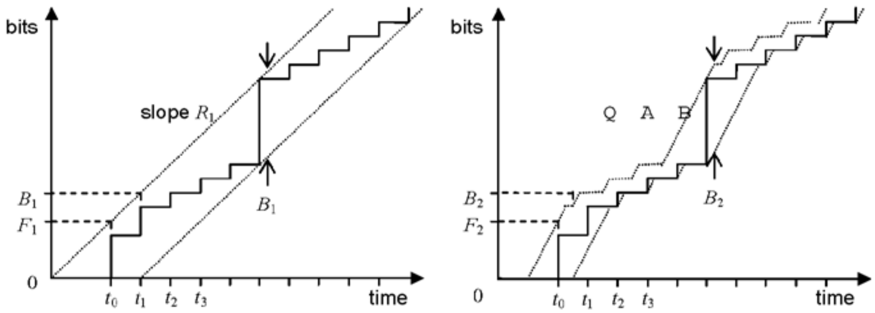

Figure C-5 – Illustration of the leaky bucket concept

If a bitstream is contained in a leaky bucket with parameters (R_1,B_1,F_1), then when it is input with peak rate R_1 to a hypothetical reference decoder with parameters $R=R_1$, $B=B_1$, and $F=F_1$, then the HRD buffer does not overflow or underflow.

C.2.2.3 Minimum buffer size and minimum peak rate

If a bitstream is contained in two leaky buckets with parameters (R_1,B_1,F_1) and (R_2,B_2,F_2), then it is also contained in any leaky bucket with parameters (R,B,F) where a) $R_1 \leq R \leq R_2$, b) $B \geq B_{min}(R)$, and c) $F \geq F_{min}(R)$ and $B_{min}(R)$ and $F_{min}(R)$ are defined by

$$B_{min}(R) = \alpha B_n + (1 – \alpha)B_{n+1}, \tag{C-26}$$

$$F_{min}(R) = \alpha F_n + (1 – \alpha)F_{n+1}, \tag{C-27}$$

and

$$\alpha = (R_{n+1} – R) \div (R_{n+1} – R_n). \tag{C-28}$$

For $R \leq R_1$,

$$B_{min}(R) = B_1 + (R_1 – R)T \tag{C-29}$$

$$F_{min}(R) = F_1, \tag{C-30}$$